IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Appellant:	Duisenberg	Patent Application	
Application No.:	10/072,358	Group Art Unit:	2181
Filed:	02/06/2002	Examiner:	Lee, Chun Kuan

For: METHOD AND APPARATUS FOR SYNCHRONIZING A SOFTWARE BUFFER INDEX WITH AN UNKNOWN HARDWARE BUFFER INDEX

APPEAL BRIEF

Table of Contents

I. Real Party in Interest

The assignee of the present application is Hewlett-Packard Development Company, L.P.

II. Related Appeals and Interferences

None. The Appellant has no knowledge of any related appeals or interferences.

III. Status of Claims

Claims 1-24 are pending. This appeal is directed to the rejection of Claims 1-24.

IV. Status of Amendments

All proposed amendments have been entered. An amendment subsequent to the Final Action has not been filed.

V. Summary of Claimed Subject Matter

Independent Claim 1 of the present application pertains to a method of processing data. Independent Claim 10 of the present application pertains to a method of processing data. Independent Claim 16 of the present application pertains to a computer system.

Claim 1: Claim 1 recites, "[a] method of processing data comprising ..." This embodiment is described at least by page 11, line 26 - page 18, line 11; flow chart 400 of Figure 4, and diagram 600 of Figure 6. With respect to Claim 1, "when a software buffer index points to a first buffer containing processed data," is described at least at page 11, lines 31 - page 12, line 16; steps 410 and 420 of Figure 4; in Figure 6; and at page 13, lines 8-23. "[S]ynchronizing said software buffer index to a hardware buffer index by sequentially searching through a plurality of buffers containing data to determine whether there is a second buffer with unprocessed data," is described at least by page 14, line 4 - page 16, line 4, and steps 440, 450, 470, 480, and 490 of flow chart 400. "[I]f there is said second buffer with unprocessed data, resetting said software buffer index to a next available buffer having processed data following said second buffer, and otherwise stopping said searching when each buffer of said plurality of buffers has been searched and a buffer with unprocessed data is not found," is described at least by page 12, lines 10-16; by page 14, line 31 - page 16, line 4; and by steps 420, 450, 460, 470, 480, and 490 of flow chart 400 in Figure 4 (process iteratively checks for unprocessed data in buffer and increments buffer during checking, and breaks out of the process and ceases searching at 450 after all buffers have been checked).

Claim 10: Claim 10 recites, "[a] method of processing data comprising ..." This embodiment is described at least by page 11, line 26 - page 18, line 11; flow chart 400 of Figure 4, and diagram 600 of Figure 6. With respect to Claim 10, "receiving an interrupt indicating data from a local area network (LAN) has been stored in one of a plurality of buffers and is ready for processing," is described at least at page 12, lines 4 - 8 and at step 410 of flow chart 400 in Figure 4. "[W]hen a software buffer index points to a first buffer containing processed data,"

is described at least at page 11, lines 31 - page 12, line 16; steps 410 and 420 of Figure 4; in Figure 6; and at page 13, lines 8-23. "[S]equentially searching through said plurality of buffers containing data to determine whether there is a second buffer with unprocessed data," is described at least by page 14, line 4 - page 16, line 4, and steps 440, 450, 470, 480, and 490 of flow chart 400. "[I]f there is said second buffer with unprocessed data, synchronizing said software buffer index to a hardware buffer index by resetting said software buffer index to a next available buffer having processed data following said second buffer, and otherwise stopping said searching when each buffer of said plurality of buffers has been searched and a buffer with unprocessed data is not found," is described at least by page 12, lines 10-16; by page 14, line 31 - page 16, line 4; and by steps 420, 450, 460, 470, 480, and 490 of flow chart 400 in Figure 4 (process iteratively checks for unprocessed data in buffer and increments buffer during checking, and breaks out of the process and ceases searching at 450 after all buffers have been checked).

Claim 16: Claim 16 recites, "[a] computer system comprising ..." This embodiment is described at least by page 9, line 26 - page 11, line 25 (computer system); page 11, line 26 - page 18, line 11 (operation); computer system 200 of Figure 2 and Figure 3; flow chart 400 of Figure 4; and diagram 600 of Figure 6. With respect to Claim 16, "a processor;" is described at least at page 9, line 33 - page 10, line 13; page 10, lines 27-29; and page 11, lines 11 - 14; and shown at least by processor 201 of Figure 2 and Figure 3. "[A] computer readable memory coupled to said processor and containing program instructions that, when executed, implement a method of processing data," is described at least at page 9, lines 22 - 29; page 9, line 31 - page 10, line 5; and shown at least in Figure 2 by processor 201 being coupled with volatile RAM 202, non-volatile ROM 203, and data storage device 204. "[W]hen a software buffer index points to a first buffer containing processed data," is described at least at page 11, lines 31 - page 12, line 16; steps 410 and 420 of Figure 4; in Figure 6; and at page 13, lines 8-23. "[S]ynchronizing said software buffer index to a hardware buffer index by sequentially searching through a plurality of buffers containing data to determine whether there is a second

buffer with unprocessed data," is described at least by page 14, line 4 - page 16, line 4, and steps 440, 450, 470, 480, and 490 of flow chart 400. "[I]f there is said second buffer with unprocessed data, resetting said software buffer index to a next available buffer having processed data following said second buffer, and otherwise stopping said searching when each buffer of said plurality of buffers has been searched and a buffer with unprocessed data is not found," is described at least by page 12, lines 10-16; by page 14, line 31 - page 16, line 4; and by steps 420, 450, 460, 470, 480, and 490 of flow chart 400 in Figure 4 (process iteratively checks for unprocessed data in buffer and increments buffer during checking, and breaks out of the process and ceases searching at 450 after all buffers have been checked).

VI. Grounds of Rejection to Be Reviewed on Appeal

1. Claims 1-2, 4-17, and 19-24 are rejected under 35 U.S.C. §103(a) as being unpatentable over the combination of Applicant's Admitted Prior Art (AAPA) in view of U.S. Patent 4,637,023 by Lounsbury et al. et al. (hereinafter "Lounsbury").

2. Claim 3 and 18 are rejected under 35 U.S.C. §103(a) as being unpatentable over the combination of AAPA in view of Lounsbury and further in view of U.S. Patent 5,860,001 to Cromer et al. (hereinafter "Cromer").

VII. Argument

1. Whether Claims 1-2, 4-17, and 19-24 are rendered obvious under 35 U.S.C. §103(a) by the combination of AAPA in view of Lounsbury.

The Rejection of 8/10/2007 indicates that Claims 1-2, 4-9 are rejected under 35 U.S.C. §103(a) as being unpatentable over AAPA in view of Lounsbury. Appellant has reviewed the cited references and respectfully submits that the claimed embodiments are not rendered obvious by AAPA and Lounsbury, either alone or in combination, in view of the following rationale.

Attention is respectfully directed to Independent Claim 1, which recites a method of processing data comprising (emphasis added):

> when a software buffer index points to a first buffer containing processed data, synchronizing said software buffer index to a hardware buffer index by sequentially searching through a plurality of buffers containing data to determine whether there is a second buffer with unprocessed data; and
>
> if there is said second buffer with unprocessed data, resetting said software buffer index to a next available buffer having processed data following said second buffer, and otherwise stopping said searching when each buffer of said plurality of buffers has been searched and a buffer with unprocessed data is not found.

Claims 2 and 4-9 depend from Claim 1 and recite further features of the claimed embodiment. Independent Claims 10 and 16 recite similar features and were rejected with the same rationale. Claims 11 - 15 depend from independent Claim 10 and recite further features of Claim 10. Claims 17 and 19 - 24 depend from Claim 16 and recite further features of Claim 16.

"As reiterated by the Supreme Court in *KSR*, the framework for the objective analysis for determining obviousness under 35 U.S.C. 103 is stated in *Graham v. John Deere Co.*, 383 U.S. 1, 148 USPQ 459 (1966). Obviousness is a question of law based on underlying factual inquiries" including "[a]scertaining the differences between the claimed invention and the

prior art" (MPEP 2141). "In determining the differences between the prior art and the claims, the question under 35 U.S.C. 103 is not whether the differences themselves would have been obvious, but whether the claimed invention as a whole would have been obvious" (emphasis in original; MPEP 2141.02(I)). Appellant notes that "[t]he prior art reference (or references when combined) need not teach or suggest all the claim limitations, however, Office personnel must explain why the difference(s) between the prior art and the claimed invention would have been obvious to one of ordinary skill in the art" (emphasis added; MPEP 2141(III)).

On page 6, line 6 - 10, the 8/10/2007 Rejection indicates that AAPA does not teach a method comprising, "if there is said buffer with unprocessed data, synchronizing said software buffer index to a hardware buffer index by resetting said software buffer index to a next available buffer having processed data following said second buffer," as recited by Claim 1. Appellant respectfully agrees with this characterization of AAPA and further submits that AAPA does not suggest or motivate such a feature. Further, Appellant respectfully submits that Lounsbury does not cure this deficiency of AAPA. More specifically, Appellant submits that Lounsbury (either alone or in combination with AAPA) does not teach or suggest or motivate, among other things, "if there is said second buffer with unprocessed data, resetting said software buffer index to a next available buffer having processed data following said second buffer," as recited by Claim 1 and similarly by Claims 10 and 16.

The Rejection (page 6, lines 14 - page 7, line 1) contends that Lounsbury teaches the above recited feature at col. 11, lines 18-41 and col. 18, lines 41-52. However, Appellant disagrees. Instead, per Appellant's understanding, the cited portions indicate that "... the buffers are checked in sequence for valid data." Appellant submits that checking for valid data is substantially different from checking for processed or unprocessed data. Further, the cited portion teaches, "If a buffer does not contain valid data, then the next buffer in the chain is checked. If a buffer does contain valid data, the blockette number which was read into the buffer along with the data is checked to see if it is the next blockette number in the blockette

number sequence. If it is, the data in that buffer is transferred to the host computer. If the blockette number is not the next number in sequence, the routine proceeds to the next buffer in the chain and checks for valid data."

Per Appellant's understanding, nothing is mentioned in Lounsbury regarding a buffer index, let alone, "if there is said second buffer with unprocessed data, resetting said software buffer index to a next available buffer having processed data following said second buffer," (emphasis added) as recited by Claim 1. Further, Appellant submits that the combination of AAPA and Lounsbury does not teach, describe, or suggest such a feature, as neither reference teaches or suggests it individually. Moreover, no explanation is provided in the Rejection as to why these differences (and overcoming them) between the claimed feature and the combination of AAPA and Lounsbury would have been obvious to one of ordinary skill in the art. Thus for these reasons, Appellant submits the combination of AAPA and Lounsbury does not render obvious the features of Claims 1, 10, and 16.

Further, "[i]f proposed modification would render the prior art invention being modified unsatisfactory for its intended purpose, then there is no suggestion or motivation to make the proposed modification" (MPEP 2143.01(V)). Moreover, "[i]f the proposed modification or combination of the prior art would change the principle of operation of the prior art invention being modified, then the teachings of the references are not sufficient to render the claims *prima facie* obvious" (emphasis added; MPEP § 2143.01(VI)).

Per Appellant's understanding, AAPA teaches a software ring buffer in which unprocessed data in the buffer is processed sequentially as it is accessed by LAN software (see, e.g., page 3, lines 21-30; page 4, lines 27-33; and prior art Figure 1 of the present application. Per Appellant's understanding, Lounsbury (col. 11, lines 18-41 and col. 18, lines 41-52) teaches a method of processing data from a buffer that does not distinguish whether data in a buffer location is processed or unprocessed, but only whether it is valid, and if valid,

whether it matches the next sequence number of data that it is seeking to process. Thus, if the sequence number associated with valid data in a buffer location matches the sequence number being searched for, then the data in the buffer is processed. Otherwise, valid data in a buffer is skipped without being processed, instead of being processed as it is accessed.

By incorporating this teaching of Lounsbury, the Rejection has created a system which can/would skip over unprocessed data in a buffer, rather than sequentially processing unprocessed data as it is accessed. Appellant submits that this combination substantially changes the principle of operation of AAPA. Appellant submits that this change in the principle of operation of AAPA is impermissible according to MPEP 2143 (cited above), and thus the use of AAPA in this fashion (even in combination with Lounsbury) is insufficient to render Claims 1, 10, and 16 *prima facie* obvious.

Therefore, for at least these reasons, Appellant respectfully submits that the combination of AAPA in view of Lounsbury is insufficient to render Claims 1, 10, and 16 obvious. Claims 2, and 4-9 depend from Claim 1, Claims 11-15 depend from Claim 10, and Claims 17 and 19-24depend from Claim 16. Thus Appellant respectfully submits that Claims 2, 4-9, 11-15, 17, and 19-24are all allowable over the rejection under 35 U.S.C. § 103(a) by virtue of their dependence upon allowable base Claims.

2. Whether Claims 3 and 18 are rendered obvious under 35 U.S.C. §103(a) by the combination of AAPA in view of Lounsbury and further in view Cromer.

The Rejection of 8/10/2007 indicates that Claims 3 and 18 are rejected under 35 U.S.C. §103(a) as being unpatentable over AAPA in view of Lounsbury and further in view of Cromer. Appellant has reviewed the cited references and respectfully submits that the claimed embodiments are not rendered obvious by AAPA, Lounsbury, and Cromer, either alone or in combination, in view of the following rationale.

Attention is respectfully directed to Independent Claim 1, which recites a method of processing data comprising (emphasis added):

> when a software buffer index points to a first buffer containing processed data, synchronizing said software buffer index to a hardware buffer index by sequentially searching through a plurality of buffers containing data to determine whether there is a second buffer with unprocessed data; and
>
> if there is said second buffer with unprocessed data, resetting said software buffer index to a next available buffer having processed data following said second buffer, and otherwise stopping said searching when each buffer of said plurality of buffers has been searched and a buffer with unprocessed data is not found.

Claim 3 depends from Claim 1 and recites further features of the claimed embodiment. Independent 16 recites similar features and was rejected with the same rationale. Claim 18 depends from Claim 16 and recites further features of Claim 16. It is appreciated that Claims 3 and 18 were rejected with the same rationale.

On page 6, line 6 - 10, the 8/10/2007 Rejection indicates that AAPA does not teach a method comprising, "if there is said buffer with unprocessed data, synchronizing said software buffer index to a hardware buffer index by resetting said software buffer index to a next available buffer having processed data following said second buffer," as recited by Claim 1. Appellant respectfully agrees with this characterization of AAPA and further submits that AAPA does not suggest or motivate such a feature. Further, as detailed above with respect to Claims 1 and 16, Appellant respectfully submits that Lounsbury does not cure this deficiency of AAPA. Moreover, upon review of Cromer, Appellant also submits that Cromer (either alone or in combination with AAPA and Lounsbury) does not teach, suggest or motivate, among other things, "if there is said second buffer with unprocessed data, resetting said software buffer index to a next available buffer having processed data following said second buffer," as recited by Claim 1 and similarly by Claim 16.

Per Appellant's understanding, nothing is mentioned in Cromer regarding a buffer index, let alone, "if there is said second buffer with unprocessed data, resetting said software buffer index to a next available buffer having processed data following said second buffer," (emphasis added) as recited by Claim 1 and similarly by Claim 16. Further, Appellant submits that the combination of AAPA, Lounsbury, and Cromer cannot teach, describe, or suggest such a feature, as none of these references teaches or suggests it individually. Moreover, no explanation is provided in the Rejection as to why these differences (and overcoming them) between the claimed feature and the combination of AAPA, Lounsbury, and Cromer would have been obvious to one of ordinary skill in the art. Thus for these reasons, Appellant submits the combination of AAPA, Lounsbury, and Cromer does not render obvious the features of Claims 1, 10, and 16.

Therefore, Appellant respectfully submits that the combination of AAPA in view of Lounsbury and further in view of Cromer is insufficient to render Claims 1 and 16 obvious. Claim 3 depends from Claim 1 and Claim 18 depends from Claim 16. Thus Appellant respectfully submits that Claims 3 and 18 are all allowable over the rejection under 35 U.S.C. §103(a), by virtue of their dependence from allowable base claims.

Conclusion

Appellant believes that pending Claims 1-24 are patentable over the cited art. As such, Appellant respectfully requests that the rejections of Claims 1-24 be reversed.

The Appellant wishes to encourage the Examiner or a member of the Board of Patent Appeals to telephone the Appellant's undersigned representative if it is felt that a telephone conference could expedite prosecution.

Respectfully submitted,
WAGNER BLECHER LLP

Dated: 1/24/2008

John P. Wagner
Registration No. 35,398
123 Westridge Drive
Watsonville, CA 95076

Phone: (408) 377-0500
Facsimile: (831) 722-2350

VIII. Appendix - Clean Copy of Claims on Appeal

1. A method of processing data comprising:

when a software buffer index points to a first buffer containing processed data, synchronizing said software buffer index to a hardware buffer index by sequentially searching through a plurality of buffers containing data to determine whether there is a second buffer with unprocessed data; and

if there is said second buffer with unprocessed data, resetting said software buffer index to a next available buffer having processed data following said second buffer, and otherwise stopping said searching when each buffer of said plurality of buffers has been searched and a buffer with unprocessed data is not found.

2. The method of processing data as described in Claim 1, wherein said synchronizing further comprises:

synchronizing said hardware buffer index and said software buffer index in response to an interrupt indicating data has been stored in one of said plurality of buffers and is ready for processing.

3. The method of processing data as described in Claim 1, wherein said synchronizing further comprises:

ignoring a first interrupt indicating data has been stored in one of said plurality of buffers and is ready for processing when said software buffer index points to said first buffer containing processed data; and

synchronizing said hardware buffer index and said software buffer index in response to a second interrupt indicating data has been stored in one of said plurality of buffers and is ready for processing when said software buffer index points to said first buffer containing processed data for a second time.

4. The method of processing data as described in Claim 1, further comprising:

determining if said first buffer contains processed data; and

processing data in said first buffer if said data is unprocessed.

5. The method of processing data as described in Claim 1, wherein said synchronizing further comprises:

wrapping around to a start buffer after searching the end buffer in said plurality of buffers when sequentially searching through said plurality of buffers, said plurality of buffers sequentially beginning with a start buffer and ending with an end buffer.

6. The method of processing data as described in Claim 1, further comprising:

when said software buffer index points to said first buffer containing processed data, using a value of said software buffer index corresponding to said first buffer as a reference value;

incrementing said software buffer index as each buffer of said plurality of buffers is searched, wherein when said software buffer index reaches one end of a range of possible values it is reset to the other end of said range; and

stopping said searching when said software buffer index reaches said reference value without finding a buffer in said plurality of buffers with unprocessed data.

7. The method of processing data as described in Claim 1, wherein each of said plurality of buffers is a local area network (LAN) buffer for storing LAN packets of data.

8. The method of processing data as described in Claim 7, wherein said software buffer index is a LAN software buffer index, and said hardware buffer index is a LAN hardware buffer index.

9. The method of processing data as described in Claim 1, further comprising:

processing said unprocessed data in said second buffer.

10. A method of processing data comprising:

receiving an interrupt indicating data from a local area network (LAN) has been stored in one of a plurality of buffers and is ready for processing;

when a software buffer index points to a first buffer containing processed data, sequentially searching through said plurality of buffers containing data to determine whether there is a second buffer with unprocessed data; and

if there is said second buffer with unprocessed data, synchronizing said software buffer index to a hardware buffer index by resetting said software buffer index to a next available buffer having processed data following said second buffer, and otherwise stopping said searching when each buffer of said plurality of buffers has been searched and a buffer with unprocessed data is not found.

11. The method of processing data as described in Claim 10, wherein said data from said LAN is a LAN packet.

12. The method of processing data as described in Claim 10, wherein a LAN driver performs said receiving, said searching and said synchronizing.

13. The method of processing data as described in Claim 10, further comprising:

determining if said first buffer contains processed data; and

processing data in said first buffer if said data is unprocessed.

14. The method of processing data as described in Claim 10, further comprising:

when said software buffer index points to said first buffer containing processed data, using a value of said software buffer index corresponding to said first buffer as a reference value;

incrementing said software buffer index as each buffer of said plurality of buffers is searched, wherein when said software buffer index reaches one end of a range of possible values it is reset to the other end of said range; and

stopping said searching when said software buffer index reaches said reference value without finding a buffer in said plurality of buffers with unprocessed data.

15. The method of processing data as described in Claim 10, further comprising:

processing said unprocessed data in said second buffer.

16. A computer system comprising:

a processor; and

a computer readable memory coupled to said processor and containing program instructions that, when executed, implement a method of processing data, comprising:

when a software buffer index points to a first buffer containing processed data, synchronizing said software buffer index to a hardware buffer index by sequentially searching through a plurality of buffers containing data to determine whether there is a second buffer with unprocessed data; and

if there is said second buffer with unprocessed data, resetting said software buffer index to a next available buffer having processed data following said second buffer, and otherwise stopping said searching when each buffer of said plurality of buffers has been searched and a buffer with unprocessed data is not found.

17. The computer system as described in Claim 16, wherein said synchronizing further comprises:

synchronizing said hardware buffer index and said software buffer index in response to an interrupt indicating data has been stored in one of said plurality of buffers and is ready for processing.

18. The computer system as described in Claim 16, wherein said synchronizing further comprises:

ignoring a first interrupt indicating data has been stored in one of said plurality of buffers and is ready for processing when said software buffer index points to said first buffer containing processed data; and

synchronizing said hardware buffer index and said software buffer index in response to a second interrupt indicating data has been stored in one of said plurality of buffers and is ready for processing when said software buffer index points to said first buffer containing processed data for a second time.

19. The computer system as described in Claim 16, wherein said method further comprises:

determining if said first buffer contains processed data; and

processing data in said first buffer if said data is unprocessed.

20. The computer system as described in Claim 16, wherein said synchronizing further comprises:

wrapping around to a start buffer after searching the end buffer in said plurality of buffers when sequentially searching through said plurality of buffers, said plurality of buffers sequentially beginning with a start buffer and ending with an end buffer.

21. The computer system as described in Claim 16, wherein said method further comprises:

when said software buffer index points to said first buffer containing processed data, using a value of said software buffer index corresponding to said first buffer as a reference value;

incrementing said software buffer index as each buffer of said plurality of buffers is searched, wherein when said software buffer index reaches one end of a range of possible values it is reset to the other end of said range; and

stopping said searching when said software buffer index reaches said reference value without finding a buffer in said plurality of buffers with unprocessed data.

22. The computer system as described in Claim 16, wherein each of said plurality of buffers is a local area network (LAN) buffer for storing LAN packets of data.

23. The computer system as described in Claim 22, wherein said software buffer index is a LAN software buffer index, and said hardware buffer index is a LAN hardware buffer index.

24. The computer system as described in Claim 16, wherein said method further comprises:

processing said unprocessed data in said second buffer.

IX. Evidence Appendix

None. No evidence is herein appended.

X. Related Proceedings Appendix

None. No related proceedings are herein appended.